Sample Image File..Size on disk: Less than 2 KB.
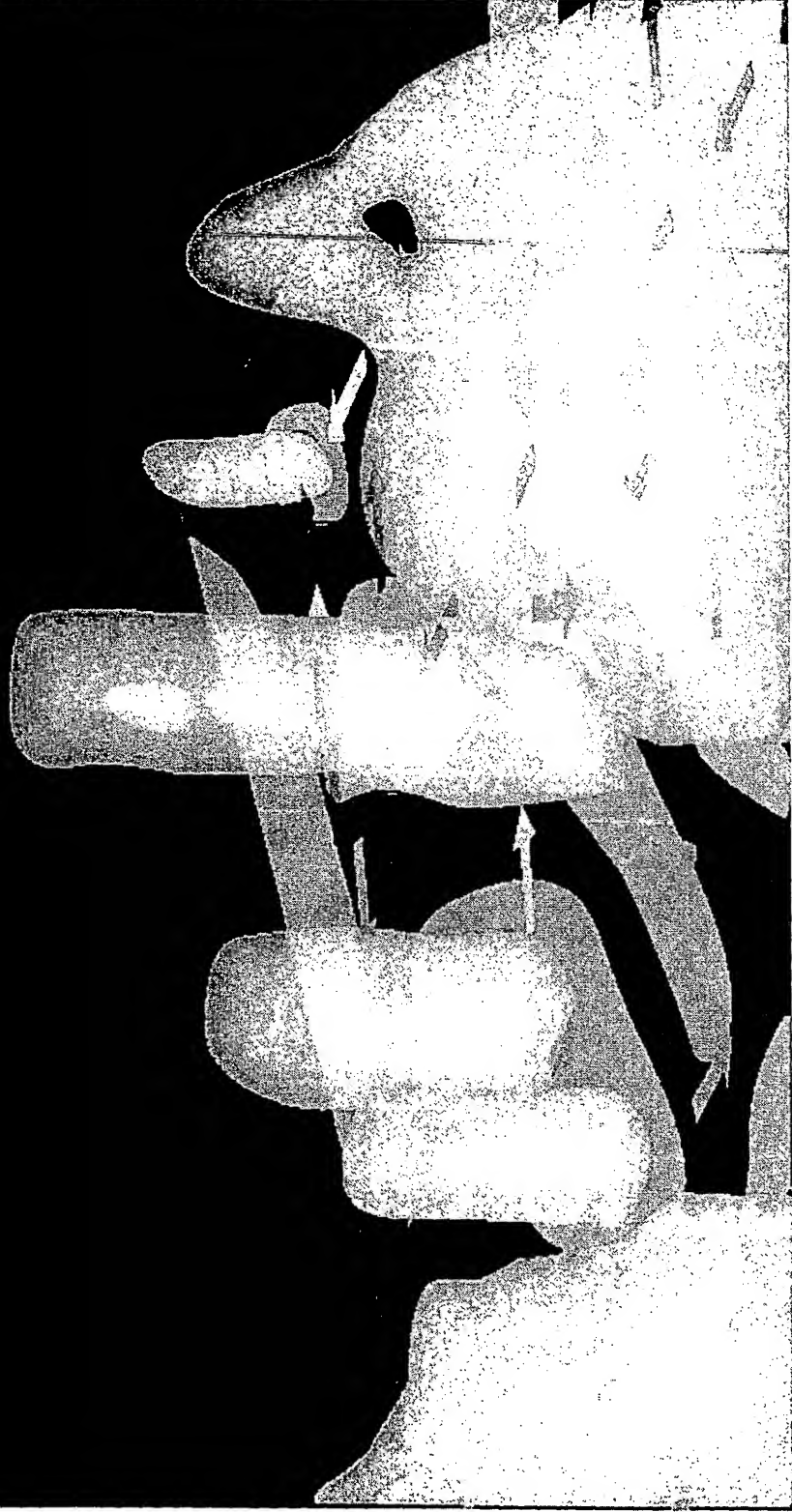
FIG. 1

## Put on Abstract Level

Analyze the image in terms of perceptual constructs of the human visual system, i.e., define it in terms of abstractions like squares, circles, gradients, rotations, etc.

— 1

## Pattern search on Abstract Level

Search for patterns on among these abstractions, in terms of which the image is now described. This is likely done ahead of time by a person. E.g., note that some arrows always bear similar structure but are rotated differently.

— 2

## Compression on Abstract Level

For a given image that is in the same 'class' as the above image, re-represent the image by describing it as a collection of parameterized patterns.

— 3

## Compression on Less Abstract Level

Take the resulting description outside of the context of abstract patterns, and look for redundancy in the description itself, e.g., look for repeated numbers.

— 4

FIG. 2

## Start

**30**

Image

**31**

Determine the class
to which an image belongs

**33**

Library of patterns
prevalent to (1 or more)
particular classes of
images--patterns are
identified a priori, thus
requiring image classes to
be fairly narrow so that
consistancy can be assured.

**32**

Isolate the patterns inherent
to this image class from this
image

**34**

Isolated image data
which constitute
examples of the
sought patterns

**35**

Describe the image data
in terms of the patterns
which they exemplify

**36**

Image data
described through
references to
patterns

*Abstract Level Compression*
*Less Abstract Compression*

**37**

Remove known forms of
redundancy found in low level
data--e.g., difference coding

**38**

Image that has
been compressed
on the abstract and
less abstract levels

FIG.3

# Encoding "Compression"

**400** 2D Image

**Start**

**401** Dismantle image into components (here, components can be requested from source)

**406** Contours represented by curve sequence and fill color

**405** Trace contours by fitting parametric curves to their borders

**404** Solid filled contours

**408** Iso-surface projections represented by curve sequence and fill color

**407** Trace Iso-surface projections by fitting curves to their borders

**403** Transparent, shaded, colored 2D projections of 3D iso-surfaces

**402** Arrow color & orientations in 3D space

**409** Represent numerical values of curve nodes as distances from one-another or local origin.

Represent numerical values of arrow colors and orientations as differences

**410** Compact border and color description of contours

**411** Compact border and color description of iso-surfaces

**412**

**413** Compact color and orientation info for arrows

**414** Text

**416** Static background image representing geography, political borders, etc.

**Start** **415**

Draw background

**417** Static description of arrow locations and skew (to make 3D tilt appearance)

Structure and definition of an arrow *(integrated into code)*

**418**

**419** Draw arrows

**420** Structure and definition of an iso-surface, including nature of gradient fill *(integrated into code)*

**421** Draw iso-surfaces

**422** Structure and definition of a contour *(integrated into code)*

**423** Draw color contours

**424** Font

**425** Draw text

## FIG. 4

# Reproduction "Decompression"

# FIG. 5

500

## 510 — Abstract Pattern Identification

**512**
Arrows
Morphologically Identical
Colors and orientations vary smoothly
Positions Constant

**514**
Background and Text
Background is static
Text is...well...text

**516**
Iso-surfaces
Amorphous
Smooth borders
Strong interior/border relation
Transparent, constant colors

**518**
Contour Lines
Amorphous
Smooth
Constant colors

## 520 — Abstract Pattern Rerepresentation

522 524 526 528

## 530 — Metadata Compression

**532**
Arrows
Colors and orientations stored as differences
Position stored in separate file

**534**
Iso-surfaces & Contour Lines
Curve points stored as difference from previous or relative to a local origin
Some curves broken into more, shorter pieces to reduce required width to store values

## 540 — Binary Output Disseminated to Clients